# Software Requirements Specification

| | |
|---|---|
| **File Name** | Photo Studio |
| **Creation Date** | September 19. 2025 |
| **Authored by** | Ringo-Lehari Purge, Martí Pujol I Saumell |
| **Reviewed by** | |

# REVISION HISTORY

| VERSION | DATE | DESCRIPTION | AUTHOR |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# CONTENTS

# 1 SW System Overview

*Specify the purpose and the overview of the SRS.*

This SRS describes requirements of a software system to be used in a Photo Studio where photos are printed and film developed.

The primary goals are to speed up the distribution of the orders, optimize employees' workflow and to give the studio administrator an overview of the work done, the income and the expenses (material used).

## 1.1 Purpose

*Describe the purpose of the system. What problem does it solve? Who are the intended users? Why is it being developed?*

One purpose of this system is to automatically create two order files (one for the customer and one for the photographer) based on the surname and desired completion date of the order to reduce receptionist's time needed for each new order.

The software system will also record orders, track their completion and the related income and expenses for logging and to give an overview to the studio administrator.

The system is meant to allow the receptionist to:
- Easily record the client's surname and desired completion time
- Mark urgent express orders.
- Automatically calculate the price (meaning the surcharge).

The system will allow the photographer to:
- submit a report on consumed materials at the end of the day
- submit the completion of an order

The system will allow the client to:
- Place an order
- Allow to view placed order
- Pay

The intended users are:
- the client
- the receptionist
- the photographer
- the studio administrator

## 1.2 Scope

*Define the scope of the system. What functionality is included? What is explicitly excluded? Mention benefits and key features.*

The system includes:
- The transmission of orders from the receptionist to the photographer.
- Price calculation based on order deadline
- The payment system
- The customer providing information to the receptionist

The system does not include:
- The developing of film/printing images.
- The transfer of printed images/developed film from the photo studio to the customer.

Benefits:
- Quick distribution of orders
- Easy reporting

Key features:

- Automatic price increase on order in case of urgent orders
- Generation of different orders for photographer and client
- Overview of income and expenses

## 1.3 Use-Case Diagram

*Provide a high-level UML use-case diagram showing main actors and their interactions with the system.*

## 1.4 General Constraints

*List technical and business constraints such as programming language, operating system, performance limitations, and standards.*

## 1.5 Assumptions and Dependencies

*State assumptions (e.g., availability of internet, supported devices) and dependencies (e.g., external APIs, hardware).*

## 1.6 Acronyms and Abbreviations

*List all acronyms and abbreviations used in the document along with their explanations.*

| Terms Used | Description of terms |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

# 2 Functional Requirements SW

## *2.1 Features / Functions to be Implemented*

*All functional requirements should be derived from User Stories or Use Cases.*

*This means that instead of listing abstract features, you first describe how users interact with the system and what goals they achieve.*

*User Stories – short, simple descriptions of a feature told from the perspective of the user (e.g., "As a registered user, I want to reset my password so that I can regain access to my account.").*

*Use Cases – structured scenarios that describe interactions between actors and the system, including preconditions, steps, and outcomes.*

*From these stories/cases, you can then identify:*

- *User interactions (e.g., authentication, profile management).*
- *Business processes (e.g., order processing, reporting).*
- *Integrations (e.g., with external APIs or third-party systems).*
- *System logic (e.g., validation, workflows, automation).*
- *Algorithms (if required, e.g., recommendation or prediction).*

*Each function must be traceable back to a User Story or Use Case, ensuring that the system is built strictly according to user and business needs.*

User stories:

- As the studio administrator I want the photographer to have an efficient workflow with as minimal overhead as possible, so that the orders are completed more quickly.
- As the receptionist I want to forward orders with accurate information so that I don't give out wrong orders.
- As the receptionist I want to have orders generated within a matter of seconds, so that the customer and photographer don't have to wait long for them.
- As the receptionist I want to automatically calculate daily income so that I don't report wrong numbers.
- As the photographer I want to have clear deadlines so that I complete orders in a sensible sequence.
- As the photographer I want to have a automatic system to calculate daily resource usage so that I don't have to keep track of it and less mistakes happen.
- As the client I want to easily place my order so that getting my photo printed doesn't take me longer than it needs to.

From these stories identifying functions:

User interactions:

- Reading user data

- Entering user data
- Entering order progress
- Placing order

- Reading an order

Business Processes:

- Order processing in a sensible sequence
- Reporting income
- Reporting completed orders
- Reporting used resources
- Payment

System logic:

- Automate prices calculation
- Automate generation of orders under five seconds
- Automate daily income sum calculation
- Automate daily resources used sum calculation

- Ensure the order has the correct deadline on it

- Client information and deadline are sent to the receptionist.

Integrations:

- Some DBMS?

## 2.1 Acceptance Criteria

*Define how each requirement will be validated: test cases, acceptance tests, or quality metrics.*

## 2.2 Implementation Requirements

*Provide details of specific implementation requirements if applicable. For example, integration with existing systems, supported platforms, or algorithms.*

# 3 SW Non-Functional Requirements

## 3.1 Resource Consumption

*Specify performance and resource limits (CPU, memory, storage, response time).*
Maximum memory usage: ≤ 100 MB
Maximum file size for daily logs: ≤ 5 MB

## 3.2 License Issues

*State licensing requirements and constraints on third-party software or libraries.*
Only standard C++ STL libraries are allowed.
No proprietary third-party libraries are permitted.
External libraries may only be used if they have permissive open-source licenses (MIT, Apache-2.0).

## 3.3 Coding Standard

*Define coding style and standards that must be followed.*
Each function and class has to include descriptive comments.
Tests have to cover all critical functions.

## 3.4 Modular Design

*Specify architectural requirements such as modularity, extensibility, and maintainability.*

The system should have separate modules for:

Placing the order
Payment
Recording customer details.
Saving details to a db along with completion?
Passing order to client.
Passing orders to photographer
Price calculator.
Photographer reporting completion.
Photographer reporting consumed materials.
Receptionist reporting completed orders & revenue.
Overview of consumed materials & revenue by studio administrator.

## 3.5 Reliability

*Define requirements for reliability, error handling, and fault tolerance.*

The system must not accept invalid input values.
Errors have to be logged to a text file.

## 3.6 Portability

*List target platforms and environments where the system should operate.*

The system must run on Linux.
Identical inputs must produce identical outputs on both platforms.

## 3.7 General Operational Guidelines

*Provide guidelines for scalability, robustness, ease of use, and maintainability.*

The system must be robust, easy to maintain, and simple to use.
All operations must be logged for accountability and auditing purposes.

# 4 SW Design Artifacts

## *4.1 CRC Cards (Class–Responsibility–Collaboration)*

*List the main classes with their responsibilities (action verbs) and collaborators (related classes); keep items concise and implementation-agnostic.*

## *4.2 Conceptual UML Diagram (entities & relationships)*

*Draw a conceptual class diagram with key entities and their relationships; focus on nouns from User Stories/Use Cases, omit methods and low-level details.*